Renderer Overview

- OpenInventor Parser
- Point Transformation and Projection
- Line Rasterizer
OpenInventor Language

PerspectiveCamera {   # Sets up Camera location, field of view, etc.
  position      0 0 1
  orientation   0 0 1 0
  nearDistance  1
  farDistance   10
  left          -1
  right         1
  top           1
  bottom    -1
}

Separator {
  Transform {
    translation  tx ty tz  # 3 real numbers
    rotation     axisX axisY axisZ angle
    scaleFactor  sx sy sz  # 3 real numbers
  }  # end of Transform

  # Creates list of 3D pts, named with integers starting at 0
  Coordinate3 {
    point [x0 y0 z0,
            ...
            xn yn zn
  ]

  # Uses the integer names of the pts to make polygonal faces
  IndexedFaceSet {
    coordIndex [
      face0point0, face0point1, ... -1,
      face1point0, face1point1, ... -1,
      ...
      faceNpoint0, faceNpoint1, ... -1
    ]
  }
}
Camera Block

• One camera for the entire scene:

```plaintext
PerspectiveCamera {
    position      0 0 1
    orientation   0 0 1 0
    nearDistance  1
    farDistance   10
    left          -1
    right         1
    top           1
    bottom        -1
}
```
Separator Block

• Organizes all geometry undergoing the same transformation

separator {
    Transform {}
    Coordinate3 {}
    IndexedFaceSet {}
}

Sunday, January 24, 2010
Transform Block(s)

Transform {

  translation tx ty tz
  rotation  axisX axisY axisZ angle
  scaleFactor sx sy sz

}

...
Coordinate3

Coordinate3 {
    point [
        x0 y0 z0,
        ...,
        xn yn zn
    ]
}

Sunday, January 24, 2010
IndexedFaceSet

IndexedFaceSet {
  coordIndex [
    face0point0, face0point1, ... -1,
    face1point0, face1point1, ... -1,
    ...  
    faceNpoint0, faceNpoint1, ... -1
  ]
}

Sunday, January 24, 2010
Transformations

Object ==> World ==> Camera ==> NDC
Transformations

Object ==> World

- Construct matrix for each transform block
  - $\text{Transform}_i = T_i R_i S_i$

- Remember: $T$, $R$, $S$ can appear in any order (or not at all!)

- Combine all of the separator’s transforms:
  - $O = \text{Transform}_0 * \text{Transform}_1 * ...$

- (First transform is applied last)
Transformations

World ==> Camera

• Use position and orientation from the PerspectiveCamera block to construct Camera ==> World matrix, TR

• Camera transform is the inverse:

  \[ C = (TR)^{-1} = R^{-1}T^{-1} \]

• Use formulas for inverted translation/rotation instead of inverting the matrix.
Transformations

Camera ==> NDC

• All coordinates in [-1, 1]

• Perspective projection using parameters from the PerspectiveCamera block

• After projection and homogenization, \((x, y)\) coordinates are used to draw lines.

• \(z\) is used in HW3 as a depth value

• Formula and derivation are linked from the HW page
Your Program

- wireframe xres yres
- (xres, yres are output image resolution)
- Read OpenIV file from stdin
- Write PPM image to stdout
- Test against the provided examples
Plan

• Define an AST for the OpenInventor Language
• (mostly complete) BNF given on the HW page
• Write your parser (translate grammar into Bison syntax)
• Add code to apply parsed transformation to points and draw using your HW1 rasterization code

Sunday, January 24, 2010
Use Left Recursion!

Instead of

```c++
singles:
    NUMBER
    {
        $$ = new list<double>;
        $$->push_back($1);
    }
    | singles COMMA NUMBER
    {
        $1->push_back($3);
        $$ = $1;
    }
;```